Docket No.: ZAY-99-029

_____ REMARKS

In an office action dated 8 April 2002, the Examiner rejects claims 1-9 (all pending claims) and objects to the specification. In response to the office action, Applicant amends the specification, cancels claims 1-9, and adds claims 10-31. Applicant also traverses the rejections. Claims 10-31 now remain in the application. In light of the added claims and the arguments set forth below, Applicant requests that the Examiner allow this application.

The Examiner specifically objects to the specification because of module on page 6, lines 11-17 and the reference numeral 40 in page 12 line 7. Applicant has amended the specification to correct these and other typographical and editorial errors found in the specification.

The Examiner rejected claims 1-2 and 4-7 as being anticipated by U.S. Patent Number 5,640,595 issued to Baugher et al (Baugher) under §102 (b). Applicant has canceled claims 1-9 and substituted claims 10-31 to overcome this rejection.

Applicant has written claim 10 to recite a method for providing a transaction layer for a module that detecting a link layer, receiving capabilities of the link layer and generating a link layer configuration. Baugher does not provide a transaction layer that provides the method recited by claim 10. Instead, the method described by Baugher (See Figs. 6A-C, and Fig. 7, described in Col. 8, line 54- Col. 14, line 41), is performed by a configuration subsystem (56) that generates the files read by a reservation subsystem (58) to determine resource allocation during multimedia data transfers. The method recited in claim 10 is different in that the transaction layer software maintains the information in order to configuration the link layer below. There is no separate use of the configuration system or of a reservation subsystem to determine configurations. This makes the link layers transparent to the layers above the transaction layers in the recited claims and minimizes the reliance of the transaction layer on the operating system and other subsystems in a module. Therefore, claim 10 is not taught by Baugher. Therefore, Applicant respectfully requests that the §102 (b) rejection be removed.

Applicant also notes that the Examiner rejected claim 8 as anticipated by U.S. Patent Number 6,247,083 issued to Hake et al. (Hake) under §102 (e). Further, the

7

Examiner rejects claims 3 as being obvious from Baugher in view of Hake under §103(a). Claim 10 claims a method for providing a transaction layer the performs detecting a link layer, receiving capabilities of the link layer and generating link layer configuration from the capabilities. Hake does not teach this. Hake teaches an IEEE 1394 serial bus which has a physical layer, a link layer, and a transaction layer. (SEE Col. 1, lines 37-41). The physical layer provides the physical connection to the serial bus to transmit packets (See Col. 1, lines 42-51). The link layers performs the digital logic to deliver and receive packets from higher layers (See, Col. 1, lines 52-57). The transaction layer is described as implementing asynchronous bus transactions. (See Col. 2, lines 6-14). Generally, the invention taught in Hake is for two or more link layer Integrated Circuits (IC) to operate together with one physical layer IC to provide bi-directional transmissions. (See Col. 2, Line 50 - Col. 3, line 46). The rest of the application relates to the connection between the physical layers and the link layers and does not mention the transaction layer (See Col. 3, line 66- Col. 5, line 44). Therefore, Hake does not teach the limitations of claim 10.

Since neither Baugher nor Hake teaches the limitations of claim 10, the combination of Hake and Baugher does not teach the limitations recited in claim 10. Further, even if the combination of Baugher and Hake teaches the limitation of claim 10, there is no motivation to combine the references. A motivation to combine teachings is required by MPEP 2143 to establish obviousness. The motivation to combine references must be explicit or implicit in the references. (See, In re Kotzab, 217 F.3d 1365 (Fed, Cir. 2000). "The mere fact that references can be combined or modified does not render the resultant combination obvious unless the prior art also suggests the desirability of the combination." In re Mills, 916F.2d 680, 16 USPQ2d 1430 (Fed. Cir. 1990). The Examiner has not provided any evidence in the prior art of any reason why one skilled in the art would want to combine these two references. Both of these references achieve the desired results alone and there is no reason in the teachings showing why one skilled in the art would want to combine the references. Therefore, the combination cannot be used against claim 10. Thus, Applicant requests claim 10 be allowed.

Claims 11-20 are dependent upon claim 10 and are therefore allowable for at least the same reasons as claim 10. Thus, Applicant requests that claims 11-20 be allowed.

Claim 21 recites a program storage device that stores instructions for performing the method of claim 10. Therefore, claim 21 is allowable for at least the same reasons as claim 21 and should be allowed.
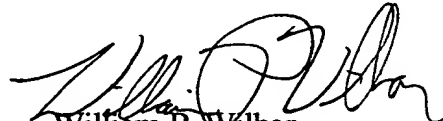
Claims 22-31 are dependent upon claim 21 and are allowable for at least the same reasons as claim 21. Therefore, Applicant respectfully requests that claims 22-31 be allowed.

If the Examiner has any questions regarding this amendment or the application, the Examiner is invited to telephone the undersigned.

Respectfully submitted,
Sierra Patent Group, Ltd.

Dated: August 6, 2002

Sierra Patent Group, Ltd.
PO Box 6149
Stateline, NV 89449
(775) 586-9500

William P. Wilbar
Reg. No. 43,265

Marked Up Version of the Changes to the Specification

<u>In the specification</u>

Please change the paragraph beginning at page 6, line 4 as follows:

The link devices [which] for which the invention provides link driver configuration may comprise the same or different types. In cases where the link devices are the same type, the invention configures the link driver for each link device according to the behavior of the link device (e.g., the type of communication carried out by the link device) as well as the capabilities of the link device. For example, in a module having first and second nodes, each node having a link device, the <u>first node</u> [module] may be configured [such that the first node carries out] <u>for</u> asynchronous communication, while the second [module] <u>node</u> may be configured for isochronous communication. Even though the link device in the first [module] <u>node</u> is identical to the link device in the second [module] <u>node,</u> the invention may provide link driver configuration optimized for asynchronous data transfer to the link device in the first [module] <u>node</u> and link drive configuration optimized for isochronous data transfer to the link device to the link device in the second [module] <u>node</u> to thereby support the behavior carried out by each respective module.

Please change the paragraph beginning at page 9, line 22 as follows:

A main pointer 14 provides a link to the link data structure 10. Link devices nodes 12a through 12n further may include a peer pointer 16a through 16(n-1) to thereby allow TNF kernel to navigate the link data structure 10 for each link device in the module. Thus, node 12a includes peer pointer 16a to node 12b, node 12b includes peer pointer <u>16b</u> to <u>node</u> 12c, etc. Since the last node does not have additional peers to point to, node 12n does not include a peer pointer, shown as null pointer 18.

Please change the paragraph beginning at page 11, line 14 as follows:

In operation, when module 26 is initialized, Link drivers 40a through 40c are loaded (initialized) for respective Links 30a through 30c. As noted above, drivers are loaded according to the type of module involved. For embedded systems, the method for installing device drivers will vary depending on the needs of the implementation. Device drivers for locally resident drivers may be precompiled into a ROM image. Under

this arrangement, at boot time the drivers would be called to perform initialization thereof. Link drivers 40a through 40c are then configured as described below. TNF kernel 38 becomes aware of LINKS 30a through 30c. For each Link 30a through 30c, the TNF kernel 38 creates link device nodes (12a through 12c) using the link data structure 10 as described above in conjunction with FIG. 2. As noted above, other data structures may be used without departing from the spirit and scope of the present invention.

Please replace the paragraph beginning at page 13, line 10 as follows:

The module 42 further includes device driver services (IO coordinator 44) operatively coupled to the TNF kernel 38 and the LINKS 30a through 30c. The IO coordinator 44 provides, among other things, [even] <u>event</u> notification to TNF kernel 38 of links 30a through 30c.

Please replace the paragraph beginning at page 14, line 15 as follows:

While the above illustrative embodiments (module 26 and module 42) were described using three nodes 28a through 28c, the method of the invention may also be carried out with a module [have] <u>having</u> one or more nodes. As illustrated, the method of the invention may be carried out with or without device driver services (IO coordinator 44, or other like messaging services).

11

In the claims

10. A method for providing a transaction layer for a module having at least one node connected to a serial bus that configures a link device for each of said at least one nodes comprising:

    detecting a link driver;

    receiving capabilities of said link driver;

    generating a link driver configuration for said link driver from said capabilities of said driver; and

    loading said link driver configuration into said link driver.

11. The method of claim 10 further comprising:

    querying said link driver for said capabilities.

12. The method of claim 11 further comprising:

    receiving said capabilities of said link driver from said link driver.

13. The method of claim 10 further comprising:

    storing said capabilities of said link driver.

14. The method of claim 13 wherein said step of storing said capabilities comprises:

    generating a node in a linked list for said link driver; and

    storing said capabilities of said link driver in a data field of said node.

15. The method of claim 10 further comprising:

    receiving configuration information for said link driver.

16. The method of claim 15 wherein said step of generating said link driver configuration comprises:

    generating said link driver configuration from said capabilities and said configuration information.

17. The method of claim 15 further comprising:

storing said configuration data.

18. The method of claim 17 further comprising:

generating a node in a linked list for said link driver; and

storing said configuration information of said link driver in a data field of said node.

19. The method of claim 10 further comprising:

receiving an input of user defined configuration data for said link driver.

20. The method of claim 19 wherein said step of generating said link driver configuration comprises:

generating said link driver configuration from said capabilities and said user defined configuration data.

21. A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to provide a transaction layer for a module having at least one node connected to a serial bus that configures a link device for each of said at least one nodes that performs a method comprising:

detecting a link driver;

receiving capabilities of said link driver;

generating a link driver configuration for said link driver from said capabilities of said driver; and

loading said link driver configuration into said link driver.

22. The program storage device of claim 21 wherein said method further comprises:

querying said link driver for said capabilities.

23. The program storage device of claim 22 wherein said method further comprises:

receiving said capabilities of said link driver from said link driver.

24. The program storage device of claim 21 wherein said method further comprises:

storing said capabilities of said link driver.

13

25. The program storage device of claim 24 wherein said step of storing said capabilities comprises:

generating a node in a linked list for said link driver; and

storing said capabilities of said link driver in a data field of said node.

26. The program storage device of claim 21 wherein said method further comprises:

receiving configuration information for said link driver.

27. The program storage device of claim 26 wherein said step of generating said link driver configuration comprises:

generating said link driver configuration from said capabilities and said configuration information.

28. The program storage device of claim 27 wherein said method further comprises:

storing said configuration data.

29. The program storage device of claim 28 wherein said method further comprises:

generating a node in a linked list for said link driver; and

storing said configuration information of said link driver in a data field of said node.

30. The program storage device of claim 21 wherein said method further comprises:

receiving an input of user defined configuration data for said link driver.

31. The program storage device of claim 30 wherein said step of generating said link driver configuration comprises:

generating said link driver configuration from said capabilities and said user defined configuration data.